26th Annual INCOSE International Symposium (IS 2016)
Edinburg, Scotland, UK, July 18-21, 2016

Interface Management for a NASA Flight Project using Model-Based Systems Engineering (MBSE)

Kevin Vipavetz
NASA Langley Research Center
Hampton, VA 23681
Kevin.G.Vipavetz@nasa.gov

Dr. Thomas A. Shull
Analytical Mechanics Associates, Inc.
Hampton, VA 23681
Thomas.A.Shull@nasa.gov

Dr. Samantha Infeld
Analytical Mechanics Associates, Inc.
Hampton, VA 23681
Samantha.I.Infeld@nasa.gov

Jim Price
Langley Research Center, NASA
(757) 864-7079
James.E.Price@nasa.gov

Abstract. The goal of interface management is to identify, define, control, and verify interfaces; ensure compatibility; provide an efficient system development; be on time and within budget; while meeting stakeholder requirements. This paper will present a successful seven-step approach to interface management used in several NASA flight projects. The seven-step approach using Model Based Systems Engineering will be illustrated by interface examples from the Materials International Space Station Experiment-X (MISSE-X) project. The MISSE-X was being developed as an International Space Station (ISS) external platform for space environmental studies, designed to advance the technology readiness of materials and devices critical for future space exploration. Emphasis will be given to best practices covering key areas such as interface definition, writing good interface requirements, utilizing interface working groups, developing and controlling interface documents, handling interface agreements, the use of shadow documents, the importance of interface requirement ownership, interface verification, and product transition.

Introduction

Background. The MISSE-X was a technology demonstration project to expand International Space Station (ISS) utilization and to advance the Technology Readiness Level of new space materials, devices, and subsystems. Experimenters were to be provided with affordable access to space by deploying an external ISS facility to host small modular technology demonstration experiments (TDEs). The MISSE-X facility was to be installed robotically and accommodate robotic on-orbit removal, return, and replacement of TDEs, which were housed in Modular Experiment Containers (MECs). Classified as a medium-sized, high-risk project, the system engineering and interface management working groups were combined, and simplified internal review processes were used. The successful MISSE-X design developed by NASA led directly to a commercially available version. The MISSE-X Systems Engineering (SE) team used the SysML plug-in within the UML tool, MagicDraw® to conduct most model-based systems engineering (MBSE), including descriptions of the interface process, and managed the actual requirements and interface definitions in the MBSE tool, CORE®.

Terminology and Basics

Interface. Per INCOSE, an interface is, "A shared boundary between two functional units, defined by functional characteristics, common physical interconnection characteristics, signal characteristics, or other characteristics, as appropriate"(Wiley 2015). An interface is a common functional or physical boundary between two systems of interest (SOIs) over which the two interact. An interface is not a system itself, nor an organizational boundary, although organizational responsibility may be associated with the interfacing SOIs and implementation agreements. As the flight project's system of interest (SOI) is developed, its external and internal interfaces need to be managed, as part of SE effort.

Interface requirement. An interface requirement, which is associated with a SOI, specifies a characteristic of an interface or a characteristic of what crosses the interface. There must be complimentary or matching requirements for both sides of the interface (the interacting SOIs).

Interface Definition. The interface definition specifies the boundary between the two sides. It is the result of the technical design process to meet the interface requirement(s). It generally takes a coordinated or negotiated effort from the two sides of the interface to develop an interface definition. To develop an interface definition, each of the interacting SOIs (each side of the interface) must know three things: the characteristics of each system at the interface (e.g., material, structure, mass, loads…); the characteristic of what is crossing the interface (e.g., current, data, strain, sheer, fluid, heat…); and what is the media of the interaction (e.g., attachment bolts, wires, pipes, environment…).

Interface Management. The goal of interface management is to identify, define, and control interfaces, and to ensure compatibility and effective system development; on time within budget, meeting stakeholder requirements. One way to ensure this is consistent and clear communication within the project and with organizations responsible for external systems with which the project's SOI must interact. The use of MBSE aids in reaching this goal efficiently.

Interface Control Working Group (ICWG). A working group to develop and control the interface management processes and the interfaces. Establishment of an ICWG is essential. The activities may include:

- identifying interfaces
- developing interface definitions
- reviewing change requests to interface requirements or definitions
- laying the groundwork for binding agreements between organizations

The group is composed of cognizant and responsible organizational representatives and subject matter experts. Attendance may depend on the agenda of each working meeting. For MISSE-X the ICWG was combined with the broader Systems Engineering Working Group (SEWG).

Interface Management Process Documentation. The project interface management process should be documented in, or summarized and referenced from, the project's Systems Engineering Management Plan (SEMP), depending on the size of the project. In the documentation, there should be an outline of the interface procedures for identifying, defining, verifying, and validating of all external and internal interfaces. The interface management process documentation should identify the people and tools to be involved in the process and how the process will correlate to the project's schedule. An example of a portion of this content from MISSE-X is provided below (more details on particular roles and milestones were also included in the SEMP):

> *The Systems Engineering Working Group (SEWG) will meet weekly to discuss and resolve interface questions and issues with people involved with either side of the interfaces in question and other subject matter experts, recording answers and decisions on the SE wiki*

blog. The System Engineering Team will use CORE®, a model-based systems engineering (MBSE) software tool for interface management. All interfaces will be captured in the model for the system, element, and subsystem level.

The Seven-Step Process

As shown in Figure 1, the NASA SE NPR (NASA Procedural Requirements) (NPR 2013) contains a general interface management process for NASA projects. This paper describes a specific implementation of this process.

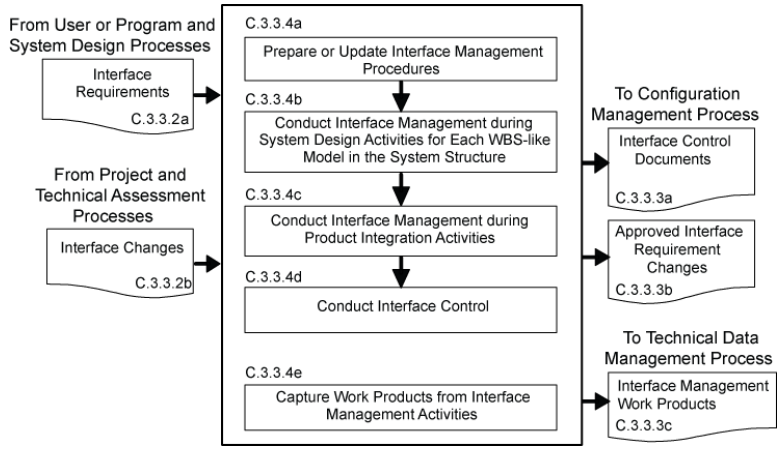


Figure 1: Interface Management Process from NPR 7123.1B

The proposed interface management process, which addresses more than the general process represented above in Figure 1, can be described in the seven major steps below. This process spans design through integration and transition to the next higher level of the architecture.

1. **Identify** – Perform interface analysis via MBSE diagrams utilizing the ICWG. Identify interface boundaries using system architecture and Concept of Operations (ConOps).
2. **Capture** – Capture interface requirements in model and/or a requirements management tool (e.g. CORE®). Assign attributes, including Requirement Owners. Place under configuration control.
3. **Define –** Develop interface design solutions and document in Interface Control Document (ICDs) or identify documents for pre-existing interfaces. Assign attributes, including agreements. Place under configuration control.
4. **Allocate** – Flow interface requirements down to the architecture level at which the hardware or software on each side will first be integrated during the realization process.
5. **Verify** – Define interface verification activities and success criteria. Conduct verification activities. Place results under configuration control.
6. **Comply** –Requirement Owners review and analyze results for compliance and approval, write verification compliance reports.
7. **Integrate** – Interconnect SOIs at their interface(s) (bring the two sides together) and validate (checkout) the integrated SOIs in the installed, operational environment, Repeat steps five to seven until project system is complete.

This seven step process has been modeled in MagicDraw® as an activity diagram shown in Figure 2. The remainder of this paper will discuss a recommended interface management methodology using MBSE within the context of this seven-step process.

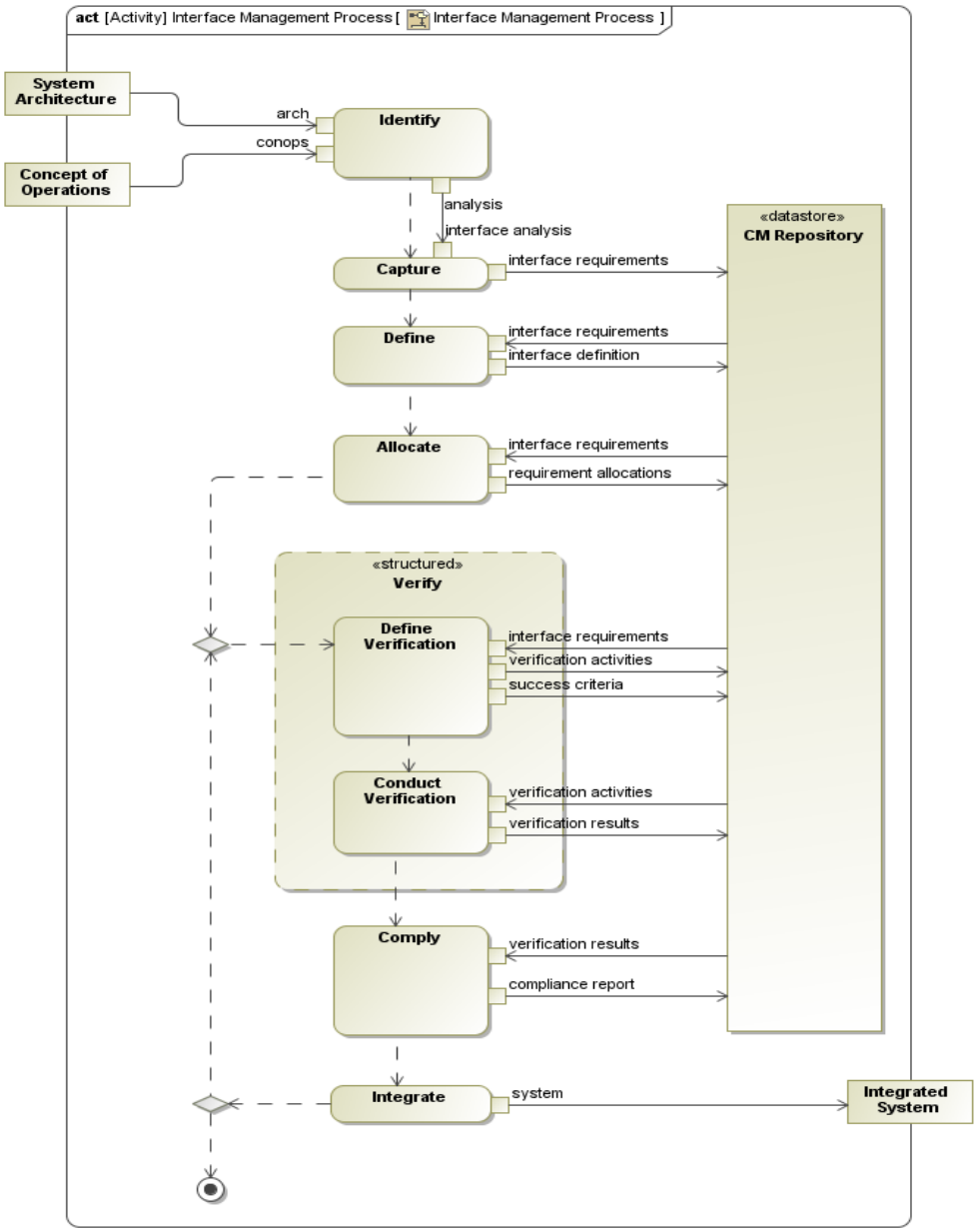


Figure 2. MBSE Activity Diagram of the Proposed Interface Management Process

1. Identify

External Interfaces. The process begins with identification of external interfaces of the project's SOI, many of which are enabling for development and mission operations. In order to completely identify enabling systems and external interfaces, use the system concept and associated ConOps coming from mission analysis and stakeholder requirements elicitation, including all major phases of the project lifecycle, to establish system boundaries. It can be helpful to ask stakeholders for scenarios of the system lifecycle, and integrate these to create the ConOps. The system boundaries should delineate where the project has control and responsibility versus where the system is interacting with entities (other systems, people, or environments) outside the project's control.

For many development projects, including MISSE-X, the mission is actually conducted within a system of systems, the project's SOI to be developed as well as the external systems. In this case, the boundaries where the project's SOI has an interface with another system in order to perform its mission can be clearly identified. The characteristic of the interface or a characteristic of what crosses the interface and any variations in the interface during different project phases should also be identified as part of this step.

For the MISSE-X development, the total mission system of systems was denoted as the MISSE-X System and the payload, associated ground assets, and external systems denoted as Elements. With this terminology, the structure was established as shown in Figure 3, an MBSE package diagram. The entities in blue represent the project's SOI, the entities over which the project has developmental or direct responsibility and control. The designated levels in the diagram refer to organizational control

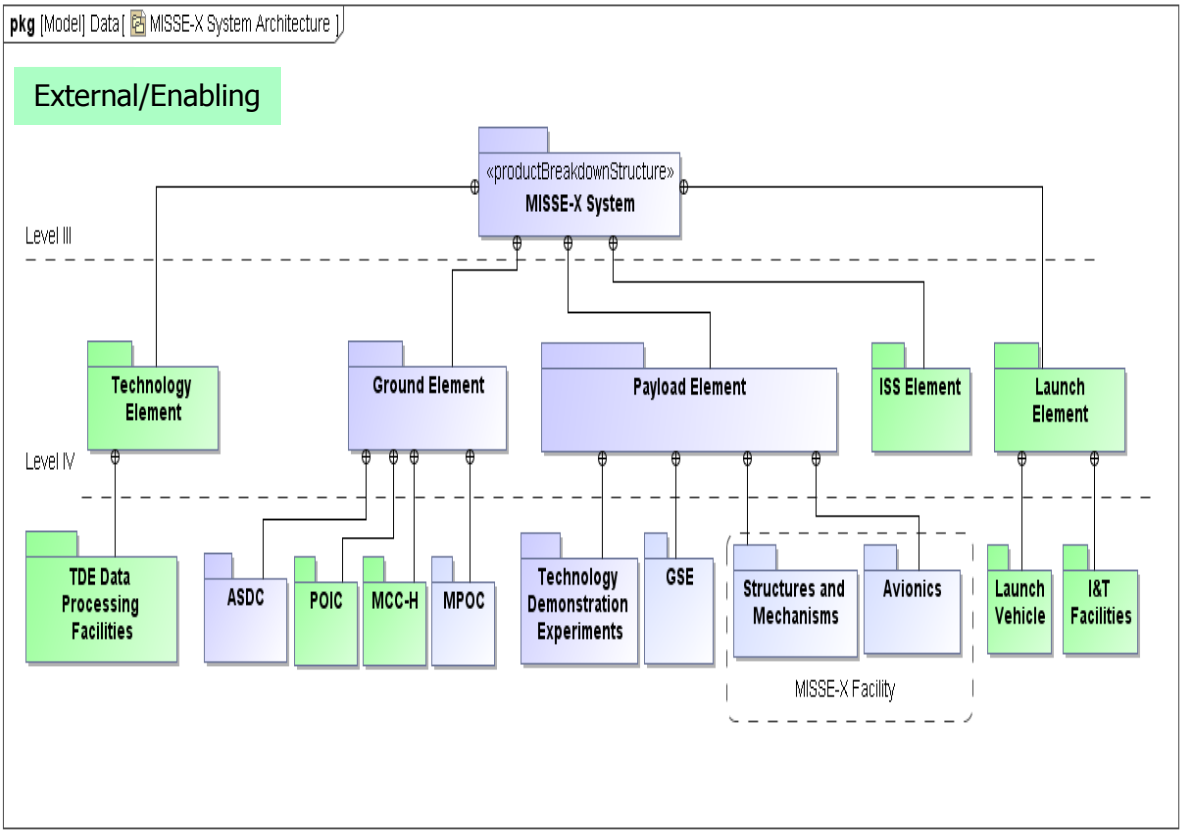


Figure 3: MISSE-X System Architecture Diagram

and architecture levels, with Level 1 representing the Agency programmatic or mission level.

For a SOI that is part of a larger space system of systems, such as an ISS payload, the external interfaces are with the enabling systems outside the project control, but possibly negotiable. Typically, there will be existing interface definitions and interface requirements for orbiting platforms, ground facilities, and launch vehicles. They could be in the form of Interface Requirement Documents (IRDs), Interface Description Documents (IDDs), Interface Control Documents (ICDs) or some other formal name. The applicable documents containing these definitions and requirements should be identified and tracked as part of the system's interface definition and subsequent management.

For example, as represented in Figure 3, the MISSE-X had an external interface with the ISS itself, the launch services and supporting integration and test facilities, and the ground systems and facilities that

service ISS and its payloads. In addition, there were interfaces throughout the integration, test, deployment, and operational activities.

For example, the mission concept included launch aboard a SpaceX Falcon 9, mounted within SpaceX's Dragon Trunk. The Facility only required survival power during launch and while awaiting removal from the Trunk, so the external interfaces with the Launch Element were mechanical, electrical, thermal, and environmental during:

- Launch site processing (Space Station Processing Facility and Launch Complex-40)
- Launch aboard the Falcon 9
- Facility delivery from Dragon trunk
- MEC retrievals and replacements from/to Dragon pressurized compartment

The MISSE-X Facility was to be mounted external to the ISS utilizing an ISS provided EXpedite the Processing of Experiments to Space Station (EXPRESS) Logistics Carrier (ELC). The Facility would obtain power, receive commands and data, and transfer data during operations, so the external interfaces with the ISS Element were mechanical, electrical, thermal, and environmental during:

- Transfers, installation, removal: Extra-vehicular robotics (EVR) (e.g. a micro-conical fixture on the Facility)
- On-orbit operations: ISS exterior ELC (e.g. interface with the ELC is through a Flight Releasable Attachment Mechanism)
- Orbital replacement units (ORU) replacement/disposal: ISS interior storage and handling
- Contingency: Extra-vehicular Activity (EVA) (e.g. Facility needs to avoid sharp edges as interface to astronauts)

All the command and communications between the Facility and the ground operations team would be provided by the ISS standard services. There were external interface to existing ISS ground facilities through the MISSE-X Payload Operation Center (MPOC).

- All data is received by the MISSE-X System from the Payload Operations Integration Center (POIC) and transferred to the MPOC
- Communications between MISSE-X and Mission Control Center at Houston (MCC-H) through the POIC
- Direct contact with MCC-H during robotic operations (installation/retrieval)

Additionally, there were external interfaces associated with the TDE, both prior to and after deployment and on-orbit exposure. The TDEs and possibly the MEC would be fabricated, assembled, integrated, and tested at the Providers facilities and delivered to NASA for installation into the Facility. Data associated with production was to be delivered to the Ground Element for review and archive. After exposure, the TDEs were removed and returned to the Provider for analysis. Reports were then delivered to the Ground Element for distribution and archive.

Internal Interfaces. Next, in order to identify internal interfaces, a system architecture definition was performed and an MBSE-based description developed using MagicDraw®. Using MBSE to define or describe the architecture allowed multiple related diagrams that defined structure (what's it made of and how's it put together), interfaces (how do you interact with it), and behavior (how does it respond to interaction). The architecture or product structure was further broken down hierarchically in a recursive manner. There is a potential interface between every pair of architectural entities at a given level. Two levels of the MISSE-X architecture are shown in Figure 3. When a project's system will be integrated into a system of systems, like MISSE-X, rather than a stand-alone satellite including launch

services, some of the elements or subsystems may have functional or communication interfaces that physically flow through external elements. For example, the MISSE-X Facility talked to the MISSE-X Ground Element through ISS Program (ISSP) resources, the ISS, MCC-H, and POIC. This complication must be taken into account. Again, variations during the entire development life cycle needs to be considered.

2. Capture

General Process. As the boundaries and interfaces are identified, one or more interface requirements should be developed on each side of the interface expressing the manner in which the SOIs need to interact or connect in order for the system of systems to be integrated, validated and allow execution of the ConOps. The requirements should address or be grouped into categories (e.g. command, data, mechanical, thermal) to simplify development, management and verification. This allows partitioning of responsibility and parallel definition, baseline, verification and closeout. Note, in our experience, "The system shall interface per document XYZ" is not a valid requirement. The word interface should not be used as a verb. Referring to an entire document imposes a huge design responsibility and verification activity, with tremendous paperwork, often leading to schedule delays.

As interface requirements are developed, they, as well as all other SOI requirements, should be captured in a requirements management (RM) tool, such as CORE®. There should be complementary or corresponding requirements for the SOI on either side of the interface. So interface requirements typically exist as pairs. The associated attributes are also included as with any product requirement. The recommended minimal set of attributes is Rationale, Trace (to parent requirement), Allocation (to child SOI), Verification Method (Test, Demonstration, Analysis, Inspection), and Owner. Assign requirement Owners for each interface requirement. This identifies who is accountable throughout the development lifecycle for definition, implementation, and compliance of the interface.

The best practice for capturing an interface definition (see step 3) is through an Interface Control Document (ICD), which should be maintained within and produced from the RM tool database or linked to the tool. When the interface definition is developed or obtained, the interface requirement will point to or be linked to the design solution. An ICD will help more fully describe that interface and will be used as the compliance measure for the interface requirement verification (see step 5). Figure 4 represents this relationship. Using the MBSE (SysML) methodology, all interface requirements can be stored in the model with all requirement attributes (CORE® terminology) or values, as in the figure.

Although the requirements baseline should be maintained in the RM tool, quite often, a System Requirements Document (SRD) is desired. These documents may go by other names, but in any case, they should be generated from the RM tool, with the generation date specified. There are several standard templates available for SRDs, but, regardless of the template used, grouping interface requirements together for ease of management is recommended. When producing a document, it is also recommended that a context diagram or SysML interface diagram be included as part of the SOI description to identify external interfaces.

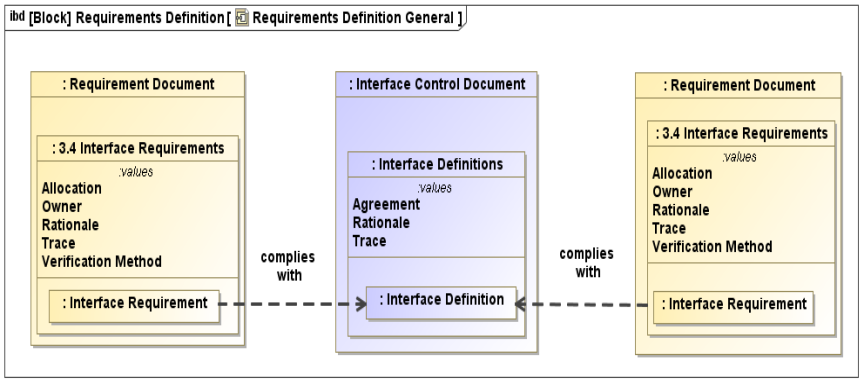
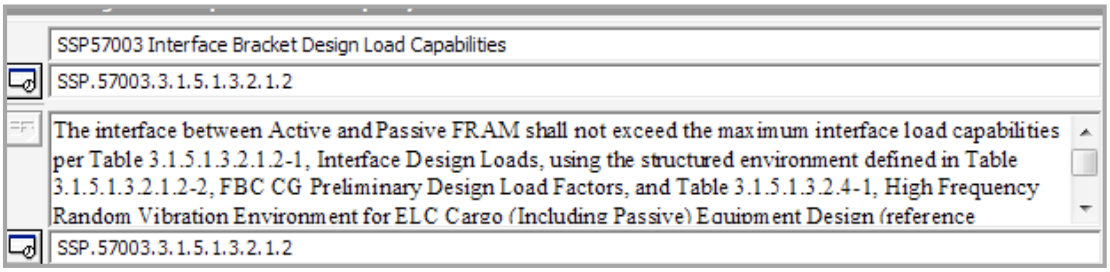


Figure 4: Interface Requirements and Definition Locations and Relationship

External System Interfaces. In the case where a project's system is interfacing with an existing system, there may be a large amount of pre-defined external interface requirements and interface definitions. For MISSE-X on the ISS, applicable sections (requirements and definitions) from the ISS Program (ISSP) documents, which contain the title prefix SSP, were entered into CORE® and treated as copies of the external side interface requirements and MISSE-X requirements as appropriate. This was done because, as is typical, requirements levied on the users of the ISS are included in the SSP documents. In fact, as is often the case, product requirements, interface definitions, verification requirements, and other applicable information is included in SSP and other existing system's documents. These project produced files (documents when exported) were called "shadow documents" because they each shadowed a particular version of a single SSP document in structure and content. This allowed the MISSE-X Project to stay current when an SSP document was undergoing a revision. The reference figures and tables from the SSP documents were also linked and put into sections of SSP "shadow documents" so that ISSP content could be accessed by the MISSE-X team as needed for design and development work.

The ISS Program negotiates and manages interfaces by producing an applicability table in the ISSP-generated ISS-Project ICD. For MISSE-X, this ICD was jointly approved by the project and the ISSP. So the MISSE-X to ISS interface requirements pointed to applicable sections of SSP documents through the filter of the applicability table. It was these referenced sections that are brought into CORE® as shown in Figure 5, which contains the SSP document reference. The figure also shows the links to applicable figures and tables.



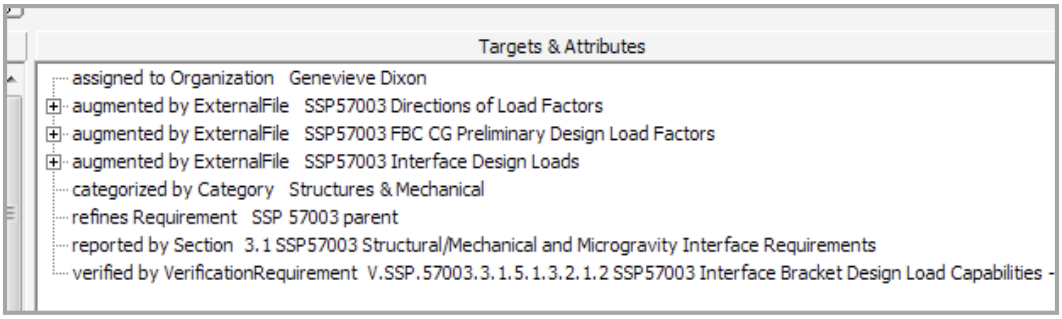


Figure 5: ISSP Interface Requirement as captured in CORE® "Shadow Documents"

The approach taken by MISSE-X was to create single interface requirements for groups of like ISS interface requirements, typically contained in a single SSP document section and likely verified together. These were assigned a single project organizational accountability. An example is shown in Figure 6.

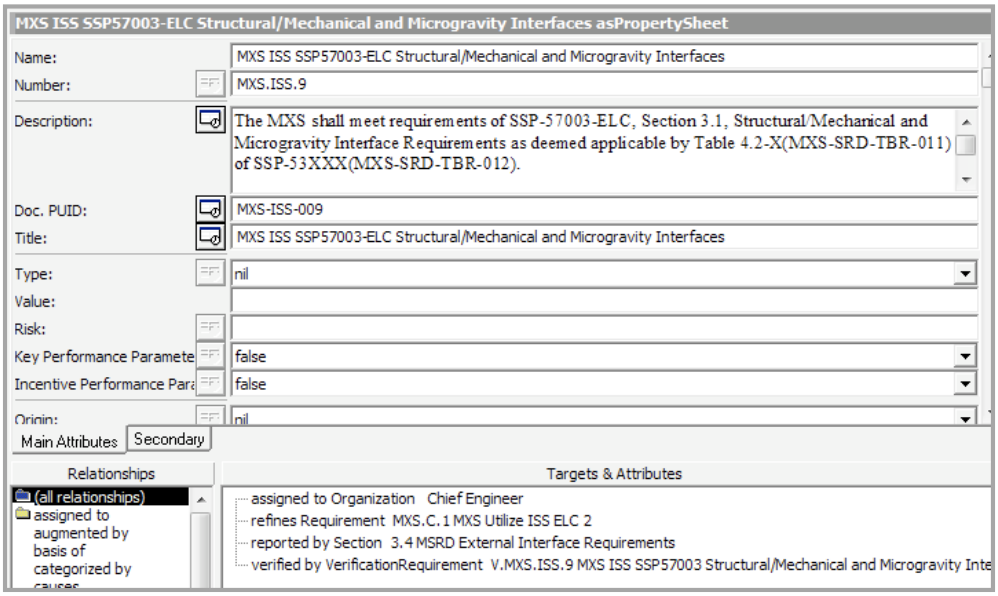


Figure 6: MISSE-X Facility Interface Requirement with ISS Captured in CORE by Section (Group)

Because of the way the ISSP manages interfaces (accommodates users) using an ICD filter (applicability table), the interface definitions are not contained in the ICD, which is ISSP-owned with project input and review, but rather, the interface definitions are actually found within various SSP documents. This is a variation or adaptation of the approach shown in Figure 4 above to that shown in Figure 7.

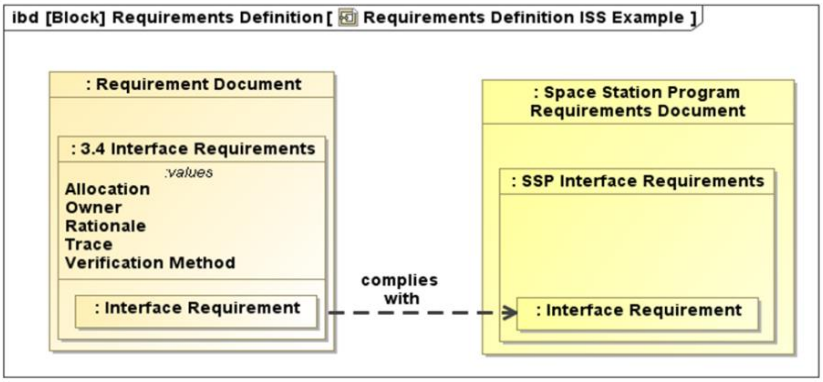


Figure 7: Interface Requirements and Definition Locations/Relationship with Existing External System Documentation

Using a template to update CORE®. To sync CORE® with an update of requirements coming externally into the project, it was useful to have a worksheet (e.g. Excel) template of an example requirement and columns for each attribute that the project was tracking in CORE®. Using a template compatible with the RM Tool allows anyone to do the majority of the updating process without being a CORE® user.

In order to use CORE® scripts to import these updates, the completed template was saved as a comma-separated values (CSV) file with a definition .txt file produced listing all attributes in the template. The "How to Guide" project in CORE® contained instructions on how to create the template and the definition file for updates. The process can be applied to any internal or external SOI requirements that are updated or as a way for project team members who are not actively updating within the model itself to submit a list of new requirements or changes for SE to import to the RM tool or MBSE model.

When SSP documents are updated between revisions, Preliminary Interface Revision Notices (PIRNs) are released by the ISSP, reviewed, and approved by affected parties. Having the SE team updating the content in CORE® as applicable PIRNs were approved meant that the "shadow documents" printed from CORE® would contain the latest approved content. This method avoided design and verification team members from having to pull SSP documents externally and individually deciding which requirements were applicable and which PIRNs affect the requirements relevant to them. This "shadow document" method can be helpful for any other external system with a large number of complex interface descriptions, requirements, and updating processes.

3. Define

Once the boundaries are identified and the interface requirements are captured, the working group and design team begin to develop detailed interface definitions that are compliant with the requirements. The definitions are not requirements, therefore no "shall" statements should be associated with interface definitions. Instead, the use of "will" for "To Be Developed" interfaces and "is" for existing is preferred. The definitions are the result of cooperative (between the interfaces sides) design activities and negotiated agreements between the responsible organizations. The definitions are the "build-to" details coming from the design process and they become the success criteria for interface verification activities. There can be multiple technical aspects to an interface definition and they can be shared at multiple architectural levels sharing the same interface (see step 4 allocation).

Definitions, like requirements, have attributes that should be captured with them. Our recommended minimal set is Rational, Trace, and Agreement. The rationales justify or provide history, like design decisions, and can point to reports or design documents. Organizational agreements, like who is providing the bolts or connectors, which sometimes do not get recorded otherwise, should be part of the definition capture. Note, these agreements may be captured in or linked to other task or contractual documents. The trace should point to the associated interface requirements from both SOI interacting at the interface boundary; the requirements using the definition as the compliance criteria. One other attribute to consider is Custodian. Assign interface definition custodians for a set of definitions or section of or entire interface document who are accountable for ensuring the interfaces are complete, following the processes, helping to track changes, and creating consensus for the interface definitions.

As mentioned previously, the definitions should be captured in the RM tool and are typically extracted as or collected in one or more ICDs, depending on the project size. An interface definition consists of one or more statements, often referring to clarifying diagrams tables, or drawings, about the characteristics of each SOI at the interface, the characteristic of what crosses the interface, or the media

involved in the interaction. For example, a Sys 1 SRD might contain the following interface requirements:

- Sys 1 shall obtain power from Sys 2 via the connections defined in ICD 2345, Drawing 3-4.
- Sys 1 shall operate on power obtained from Sys 2 having the characteristics defined in ICD 2345, Table 3.6

Conversely, the Sys 2 should contain the following interface requirements:

- Sys 2 shall provide power to Sys 1 via the connections defined in ICD 2345, Drawing 3-4.
- Sys 2 shall supply power to Sys 1 having the characteristics defined in ICD 2345, Table 3.6

Where the Sys 1 to Sys 2 ICD 2345 defines the boundary between them.

- Drawing 3-4 in the ICD contains the mating connectors, pin assignments, and grounding information in order to obtain/provide power
- Table 3.6 in the ICD contains power characteristics such as voltage, current, noise, filtering, etc.

For a project using MBSE, all interface definition statements can also be imbedded in the model with rationales (MISSE-X used the Interface element in CORE®), and traced to the interface requirements and the architectural components (Component element in CORE®) on each side of the interface. Drawings and diagrams that are part of the definition are linked to the interface definition in the model (as External File elements linked to the Interface elements in CORE®). Use of a model for interface management means that the interface can be negotiated and reviewed in the model, with informal documents capturing relevant parts in different formats, or in formal ICDs generated from the model. For example, one could print all requirements, their verification requirements, and interface definitions about a particular architectural component; or a table of interface requirements, their priorities, and the status of their verification.

4. Allocate

Interface requirements should be allocated to lower levels of the system architecture, just like any other requirement, until they are at a level that can design and implement the definition and verify the interface requirement. Keep in mind that lower levels of the architecture can interact with external interfaces at the system boundary. For the same reason, interface requirements for a lower level SOI can point to interface definitions captured at a higher level. For example, a subsystem may actually implement an element power interface, a printed circuit card may actually contain the payload's communication coaxial connector, or maybe it implements the communication protocol.

For MISSE-X, the system interface requirement in Figure 5 was addressed by looking at each applicable requirement specified. This requirement about bracket loads between the active and passive Flight Releasable Attachment Mechanism (FRAM) would be allocated, through the System-ISS interface requirement for structures and mechanisms, to the structures and mechanisms subsystem, which contains the active FRAM. The SSP requirement was allocated by ISSP on the other side of the interface to the payload platform on the ISS ELC deck that contains the passive FRAM. The interface requirement for the structures and mechanisms subsystem is the one that is under the control of the project, and will be verified through analysis of the structural design of the system against the load factors and environment data given in the SSP document.

5. Verify

The next step in the interface management process is to define how to verify the interface requirements on both sides. First the verification method must be defined: Analysis, Test, Demonstration, or Inspection. Then the specific activity should be defined, including conditions and enabling GSE SOIs and equipment, and carried out at the lowest allocated level. The levels above, up to the original interface requirement, can be verified by analysis or inspection of verification documentation from the level(s) below or verification repeated to assure nothing was affected by integration. They may also be validated at higher levels (see step 7 Integrate).

It is important to have established accountability for each set of interface definitions or ICD (Custodian, see step 3) and requirement (Owner, see step 2). Note, accountability is different from responsibility. The entire development team is responsible for system development; only one person is accountable for each requirement (the Owner) and interface control document (the Custodian).

For interfaces with existing external systems, even those allocated several levels before being implemented, the verification method and process may be pre-defined or imposed. For example, Figure 8 illustrates a verification requirement for the same ISS interface requirements about bracket.

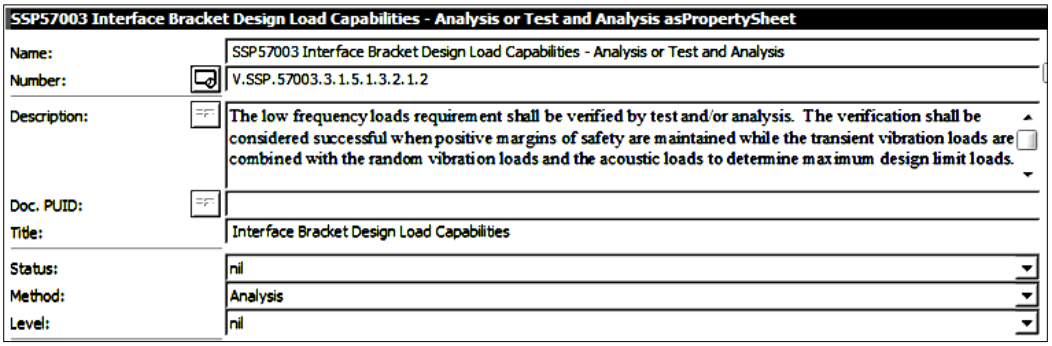


Figure 8: Example of MISSE-X Verification Requirement Extracted from SSP Documents

The system interface requirement will be verified by analysis at the structures subsystem level and by test at the Payload level with the corresponding system level interface requirement verified by analysis of the payload-level vibration test results. This is consistent with the verification method and success criterion given in the SSP document.

6. Comply

Each interface requirement must have a corresponding verification requirement in the RM tool or model containing: success criteria for meeting the interface requirement, a description of the verification activity, and a status of verification. The general success criterion for verifying an interface requirement is that the interface definition has been formally agreed to and compliance to the definition has been shown (with stated margin) by the results of the verification activity. Figure 8 above is an example of a verification requirement.

Interface requirement Owners should ensure completion of verification activities and concur by sign off that the interface complies with the interface definition. If there is any discrepancy, the owner must bring it up through SE for investigation and possible negotiation with those responsible for the other side of the interface. Note, it is not necessary for both sides to complete their interface verification to begin one side's compliance.

"Sign off" or "close-out" must be documented. For example, a project may produce a Verification Compliance Sheet (VCS) for each requirement and put each under configuration control. The set of all completed VCSs at any time would make up a Verification Compliance document (VCD) for each

SOI, which may be approved incrementally or periodically to formally record the verification status for reviews or at integration milestones. This suggested process is illustrated below in Figure 9.

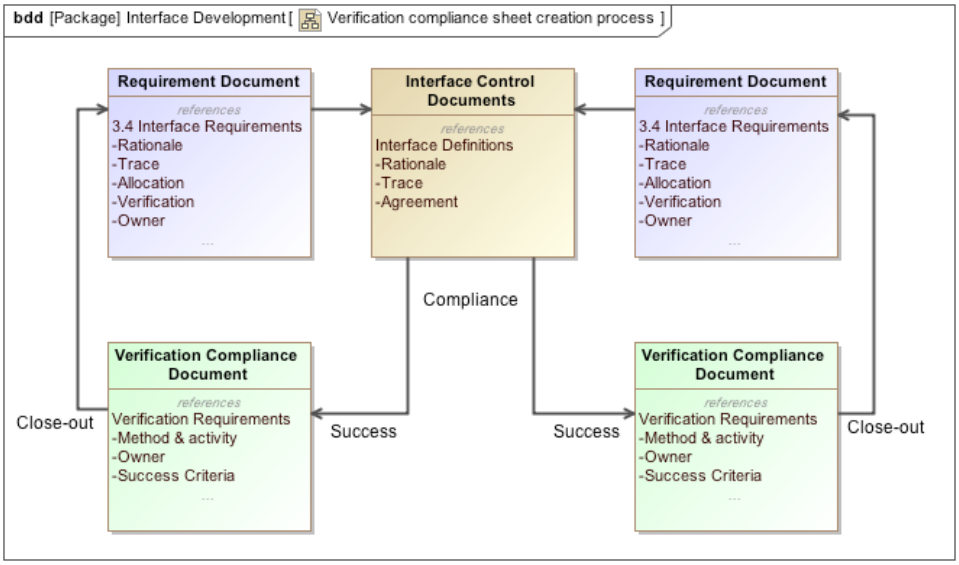


Figure 9: Verification Compliance Sheet Creation Process

The VCSs for a set of verification activities that will be done together, or owned by one person, can be produced by CORE® at one time for convenience during verification activity if all requirements and their verification requirements are part of the CORE® model. The method used by MISSE-X was to create an easy to read template in Microsoft Excel containing the following columns: verification requirement number and title, owner, description, and status, along with the three columns used by CORE® in the CSV scripts (class, folder, and name) as hidden columns. Then the Generate CSV script was used to print a file containing the set of verification requirements desired, and the resulting .csv output pasted into the .xls template unformatted. The verification requirement Owner can change the status to complete as the activities are completed and success criteria confirmed. When this data was read back into CORE® using the Advanced CSV Parser script, indicating that the owner has formally "signed off" on those verification requirements and their status changed in CORE®. Each requirement can be considered officially verified when the project approved the version of the VCD containing that requirement.

7. Integrate

After compliance on the two sides of an interface has been shown, the project can then perform integration of the two SOIs into the parent SOI at the next higher architectural level and begin verification of the parent. As part of the integration activities, SOIs are often mated with simulators to validate or confirm the interfaces before actual integration. This can be viewed as representing acceptance and transition to the next level. Following integration, the requirement Owners at the next higher level can begin verification activities to assess compliance at that level. If the implemented interface is with an external system, the verification rolls up to the system level.

This Verify-Comply-Integrate cycle is continued until the system level is reached, where system verification is performed. The suggested VCSs and VCDsthe ingrained paradigm, would be closed out and approved from the bottom up. When the top level system requirements have been verified, all the interfaces can be considered complete. This flow is represented in Figure 10.

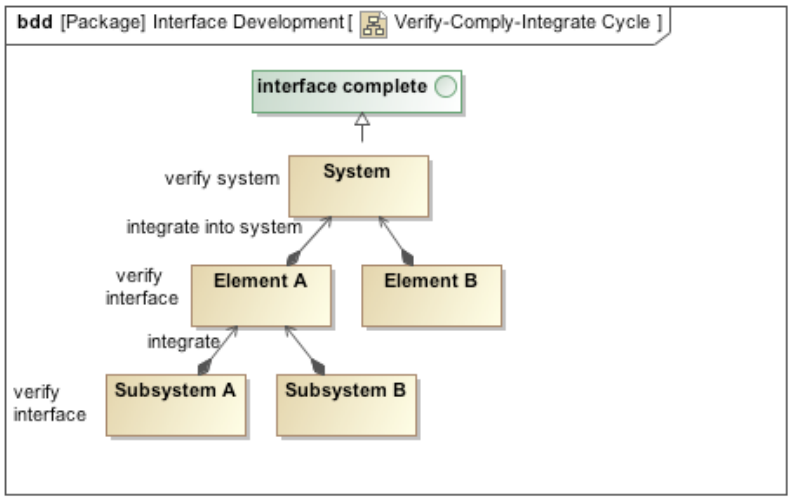


Figure 10: Verify-Comply-Integrate Cycle

Conclusion

A seven-step interface management process that spans the entire development life cycle was presented, based on the basic precept that interface requirements are distinct from interface definitions. Appropriate capture and configuration management of interface requirements and definitions allows effective system development, verification, and integration. The use of shadow documents when dealing with existing external systems helps incorporate existing documentation into the process. Assignment of requirement owners and interface custodians ensures focus. The seven-step process can improve the probability of success.

Although this process is still highly document-centric, the ingrained paradigm, the use of MBSE methodology for managing the interfaces and producing documents for review is key to efficient tracking and clear communication. It also paves the way for a less document-centric approach, in which the verification and review artifacts exist solely as model elements, and may be reviewed through model views rather than formal documents.

References

NPR, N. (2013). 7123. 1 B—NASA systems engineering processes and requirement.

Wiley (2015). INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, John Wiley & Sons.

Biography

Mr. Kevin Vipavetz is a Senior Systems Engineer in the Systems Engineering and Engineering Methods Branch, Engineering Directorate at NASA Langley Research Center. He has received over fifty awards over his thirty two years at NASA and was the Ares I-X Requirements and Verification Manager in NASA's Constellation Program. Ares I-X was the first stage prototype to replace the Space Shuttle. In 2009, Ares I-X was Time Magazine's Invention of the year and in 2010 won NASA's prestigious award for Excellence in Systems Engineering. Mr. Vipavetz has a B.S. Degree in Applied Physics with a minor in Computer Science.

Dr. Thomas Shull is a senior system engineering and management consultant since retiring from NASA. There he worked 34 years in the development of advanced technology and flight hardware and systems, primarily associated with instrumentation and data systems. He served as electronic subsystem lead, technical manager, project manager, line supervisor, middle manager and now serves as senior systems engineer, instructor, and consultant. He has a PhD from Old Dominion University in Electrical Engineering, in the area of signal processing and is a licensed Professional Engineer in Virginia and INCOSE CSEP.

Dr. Samantha Infeld was the deputy systems engineer for MISSE-X, specializing in the CORE model development. She has more recently been developing the concurrent engineering center at NASA Langley, bringing MBSE practices and tools into the center, the Engineering Design Studio. She has a PhD from Stanford University in Aeronautics and Astronautics, in the area of optimization, and worked as a systems engineer on mission concepts at NASA JPL, prior to joining AMA to be a part of NASA Langley projects.

Mr. James Price has worked for NASA for 33 years with the last 20 years being in Systems Engineering. During that time he served at multiple NASA centers and at the Jet Propulsion Laboratory on numerous spaceflight, aeronautics and technology development projects. He currently serves as the Branch Head for the Systems Engineering and Engineering Methods Branch of the Engineering Directorate at the Langley Research Center.